PHP配置文件指令：

PHP_INI_*的定义

PHP_INI_ALL:可在 任何地方设置

PHP_INI_USER：可在用户的PHP脚本与Windows注册表中设置

PHP_INI_PERDIR：可在php.ini,.htaccess,httpd.conf中设置

PHP_INI_SYSTEM：可在php.ini,httpd,conf中设置

php.ini only：只能在php.ini中设置

配置选项

Regisyer_globals(全局变量注册开关)

该选项在设置为on的情况下，会直接把用户GET、POST等方式提交上来的参数注册成全局变量并初始化值为参数对应的值，使得提交参数可以直接在脚本中使用。

Allow_url_include(是否允许包含远程文件)

可以直接包含远程的文件，在$var可控的情况下，可以通过include($var)执行php代码

Magic_quotes_gpc(魔术引号自动过滤)

在**GET，POST，COOKIE**三个超全局变量中的**单引号，双引号，反斜杠，空字符**前自动加反斜杠

Magic_quotes_runtime(魔术引号自动过滤)

对**数据库，文件**中获取的数据里的**单引号，双引号，反斜杠，空字符**前自动加反斜杠

有部分函数不受其影响

Magic_quotes_sybase(魔术引号自动过滤)

会覆盖_gpc配置选项

对**GET，POST，COOKIE**三个超全局变量中的**空字符**进行转义，并将单引号转为双引号

Safe_mode(安全模式)

文件操作函数只能操作同一用户在同一目录或配置中目录的文件

命令执行函数会报错，如popen(),system(),exec()

open_basedir php可访问目录

限制php只能访问哪些目录

Disable_functions(禁用函数)

用于禁止一些函数的使用，在php.ini文件中设置，用逗号分隔函数名

Display_errors error_reporting错误显示

display errors 表明是否显示PHP脚本内部错误的选项

以下为错误显示的级别：

- 1　E_ERROR　报告导致脚本终止运行的致命错误
- 2　E_WARNING　报告运行时的警告类错误（脚本不会终止运行）
- 4　E_PARSE　报告编译时的语法解析错误
- 8　E_NOTICE　报告通知类错误，脚本可能会产生错误
- 2047　E_ALL　报告所有的可能出现的错误

代码审计工具 掌握seay特点，其他的掌握名字

Seay源代码审计系统(C#,针对PHP):

（1）一键自动化白盒审计，新建项目后，在菜单栏中打开“自动审计”即可看到自动审计界面。点击“开始”按钮即可开始自动化审计。

（2）代码调试，代码调试功能极大地方便了审计师在审计过程中测试代码。

（3）正则编码，Seay源代码审计系统集成了实时正则调试功能，

（4）自定义插件及规则，seay源代码审计系统支持插件拓展

除此外还有 **FortifySCA,RIPS,FindBugs,Codescan**等

漏洞验证工具 掌握功能

Burp suite

Proxy功能

核心功能，代理抓包

Intruder功能

用于暴力破解

Repeater功能

用于重发测试

浏览器拓展

hackerbar

点击Load URL即可从Firefox 地址栏获取当前URL，点击Execute 之后即可发送设置好的请求数据。

firebug

可以修改HTML,CSS元素，可以嗅探请求与响应包

Modify

可以全局修改HTTP请求头

常见代码审计四种思路

1. 根据敏感关键字或函数回溯参数传递过程
2. 查找可控变量，正向追踪变量传递过程。
3. 寻找敏感功能点，通读功能点代码。
4. 直接通读全文代码

敏感函数回溯参数过程

优点：只需搜索相应敏感关键字，即可以快速地挖掘想要的漏洞，具有可定向挖掘和高效、高质量的优点。

缺点：由于没有通读代码，对程序的整体框架了解不够深入，在挖掘漏洞时定位利用点会花费一点时间，另外对逻辑漏洞挖掘覆盖不到。

espcms操作

双击定位审计后的可疑代码到该行

选中变量可以看到变量传递过程，点击跳到传递过程处

选中函数可以跳到函数主体

可以全局搜索类的实例化

漏洞本身为没有任何过滤的SQL注入

通读全文代码需掌握的关键字和功能

- 函数集文件，通常命名中包含**functions 或者 common** 等关键字，这些文件里面是一此公共的函数，提供给其他文件统一调用，所以大多数文件都会在文件头部包含
- 配置文件，通常命名中包括 **config** 关键字，配置文件包括 Web 程序运行必须的功能性配置选项以及数据库等配置信息。从这个文件中可以了解程序的小部分功能，
- 安全过滤文件，安全过滤文件对我们做代码审计至关重要，关系到我们挖掘到的可疑点能不能利用，通常命名中有 **filter、safe、check** 等关键字。
- **index 文件**，**index**是一个程序的入口文件，所以通常我们只要读一遍index文件就可以大致了解整个程序的架构、运行的流程、包含到的文件

阅读全文的优点和缺点

优点:可以更好地了解程序的架构以及业务逻辑，能够挖掘到更多、更高质量的逻辑漏洞

缺点:花费的时间比较多，如果程序比较大，读起来也会比较累

根据功能定向审计

- 文件上传功能
- 文件管理功能
- 登录认证功能
- 找回密码功能

bugfree重装漏洞代码看懂代码

BugFree 安装文件在 install\index.php，代码如下：

```
<?php
require_once('func.inc.php');

set_time_limit(0);
error_reporting(E_ERROR);
// 基本路径
define('BASEPATH', realpath(dirname(dirname(__FILE__))));
// upload path
define('UPLOADPATH', realpath(dirname(dirname(dirname(__FILE__)))).
   DIRECTORY_SEPARATOR.'BugFile');
// 配置样本文件路径
define('CONFIG_SAMPLE_FILE', BASEPATH . '/protected/config/main.sample.
    php');
// 配置文件路径
define('CONFIG_FILE', BASEPATH . '/protected/config/main.php');
```

看这段代码首先包含了 'func.inc.php' 文件，跟进这个文件可以看到一些读取配置文件、检查目录权限以及服务器变量等功能的函数，下方则是定义配置文件的路径，继续往下走，真正进行程序逻辑流程的地方如下代码所示：

```
$action = isset($_REQUEST['action']) ? $_REQUEST['action'] : CHECK;
if(is_file("install.lock") && $action != UPGRADED && $action != INSTALLED)
{
   header("location: ../index.php");
}
```

这段代码存在一个逻辑漏洞，首先判断 install.lock 文件是否存在以及 action 参数值是否升级完成和安装完成，如果是，则跳转到程序首页，这里仅仅使用了

```
header("location: ../index.php");
```

并没有使用 die() 或者 exit() 等函数退出程序流程，这个跳转只是 HTTP 头的跳转，下方代码依然会继续执行，这时候如果使用浏览器请求 install/index.php 文件则会跳转到

sql注入的利用方式

SQL 注人的攻击方式有下面几种:

- 在权限较大的情况下，可以直接写入 webshel，或者直接执行系统命令等。
- 在权限较小的情况下，可以通过注入来获得管理员的密码等信息，或者修改数据库内容进行一些钓鱼或者其他间接利用。

宽字节注入

出现：在使用PHP连接MySQL的时候，当设置“set character_set _client=gbk”时会的导致一个编码转换的注入问题

产生原理：由于单引号被自动转义成' ，前面的%df和转义字符\反斜杠(%5c)组合成了%df%5c，也就是“誫”字，这时候单引号依然还在，于是成功闭合了前面的单引号。

关于宽字节注入漏洞几种推荐方法

1)在执行查询之前先执行SET NAMES'gbk'，character-set-client=binary设置character-set -client为**binary**

2)使用mysql-set-charset(gbk)设置编码，然后使用mysqL realescape string()函数被参数过滤

3)使用**pdo方式**，在PHP5.3.6及以下版本需要设置setAttribute(PDO::ATTREMULATE PREPARES,false);来禁用prepared statements 的仿真效果

更推荐一三

二次解码urldecode注入

原理是提交参数到 WebServer 时，WebServer 会自动解码一次，若php脚本中额外又有urlcode或者rawurldecode函数来解码，则可以绕过魔术引号。

利用方法为对单引号进行两次url编码后传参

SQL注入的防范

配置中选项

magic_quotes_gpc与magic_quotes_runtime

对int型不起作用

多种过滤函数和类

- addslashes函数：检查函数的参数（必须为string类型），一般在程序的入口使用。函数先判断是否开启GPC，若未开启，则对$POST,$GET等变量进行覆盖
- mysql_[real]_escape_string：函数对字符串进行过滤，过滤**空字符，换行符，反斜杠，单双引号**
- intval等字符转换：将变量换成int类型

XSS（跨站脚本攻击）

在访问恶意页面时触发恶意脚本进行攻击

挖掘经验：在输出函数中寻找未被过滤的参数

防范：特殊字符HTML实体转码，标签事件的黑白名单

CSRF(跨站请求伪造)

利用用户的cookie等特征伪造用户进行操作

挖掘经验：

黑盒：看非静态操作的页面有没有token与referer的验证

白盒：找代码里核心文件里面有没有验证token和referer相关的代码

防范：增加token验证（注意看懂P86的代码），增加验证码

文件操作漏洞

掌握函数include()、include once()、require()和require once()的区别

远程文件包含的截断方法

- %00截断：受GPC影响，5.3后被修复
- 多个.和/截断：5.3后被修复
- 远程文件包含时利用问号(?)伪截断

文件上传漏洞

字面意思

挖掘经验：敏感函数回溯，直接找move_uploaded_file()

绕过方法：

黑名单扩展名：不常用拓展名、截断文件名、利用php与系统获取拓展名不一致（".php "注意这里有个空格）

文件头，content-type绕过：GIF89a,抓包改包

代码执行函数使用方法和变量执行过程

1. eval和assert函数 内部有无可控参数
2. preg_replace函数的\e会将第二个参数(replacement)当做代码执行
3. 动态函数,如$_GET["xx"]($_Post["xx"]);
4. 调用函数过滤不严 对于call_user_func()，array_map()等，可以通过控制参数来调用目标函数

防范

通过正则的白名单，看懂P108的正则

采用参数白名单过滤，在可预测满足正常业务的参数情况下，这是非常实用的方式，这里的白名单并不是说完全固定为参数，因为在 eval()、assert() 和 preg_replace() 函数的参数中大部分是不可预测一字不差的，我们可以结合正则表达式来进行白名单限制，用上面的 thinkphp 来举例，如果我们事先已经知道这个 URL 里面的第二个参数值由数字构成即可满足业务需求，则可以在正则里用 \d+ 来限制第二个参数内容，这样相对更加安全，用代码举例更加清晰易懂，代码如下：

```
<?php
preg_replace('/(\w+)\|(.*)/ie', '$\\1="\\2";', $_GET['a']);
?>
```

这段代码是有问题的，只要提交 /1.php?a=b|${@phpinfo()} 即可执行 phpinfo() 函数，这时候如果我们知道 \\2 的值范围为纯数字，只要正则改成 (\w+)\|(\d+) 即可解决执行代码的问题，这只是一种修复方案，最好的方法是：在 $\\1="\\2" 这里不要用双引号。

挖掘经验：敏感函数回溯，如system(),exec()，反引号等，其中注意反引号调用的是shell_exec()

防范：escapeshellcmd()过滤整条命令，将特殊字符转义

escapeshellarg()过滤参数，将参数限制在一对双引号里，会参数中原有的双引号替换为空格，以确保参数为一个字符串。

变量覆盖漏洞

利用自定义的参数值替换原有的变量值

挖掘经验：extract()函数与parse_str()函数,$$关键字

extract()：将数组注册为变量（注意PHP中的数组为键值对）

只有第二个参数为EXTR_IF_EXISTS,EXTR_OVERWRITE或缺省是才会覆盖

parse_str()函数的作用是解析字符串（格式为"a=1"）并且注册成变量，**在注册变量之前不会验证当前变量是否已经存在，所以会直接覆盖掉已有变量。**

import_request_variables():把GET、POST、COOKIE的参数注册成变量，可用在register globals 被禁止的时候。

$$关键字：**看懂P119**

逻辑处理漏洞

等于或存在判断

in_array()：比较前自动做类型转换，若数组里为数字，则数字开头的字符串会转化为数字

is_numeric()：对十六进制直接判断为true

双等于与三等于：双等于不判断类型，三等于判断类型

代码分析

看懂P128 Ecshop

绕过小技巧

$_SERVER，$_FILE不受GPC保护

编码转换都可能导致类似于宽字节注入的问题

trim()函数传入数组会报错，通过报错获取web路径

%00截断

iconv()函数遇到不能处理的字符时，后面的字符串都不会被处理，也可以用于截断

php://输入输出流的作用

php://input 读取POST提交上来的原始数据

php://output：将流数据输出

php://filter：对文件进行读写

/resourece指定数据源（/resource="example.txt"）

/read指定过滤方式（/read=string.rot13）

php代码解析标签

1)脚本标签:<script language="php">...</script>，类似于javascript

2)短标签:<?...?>，使用短标签前需要在php.ini中设置short_open_tag-on,默认是on状态。

3)asp标签:<%...%>，在PHP3.0.4版后可用，需要在 php.ini中设置asp_tags-on.默认是off。

fuzz

fuzz指的是模糊测试，对目标输入大量不同的参数来找出bug

正则表达式

注意用^与$限定开始与结束

特殊字符需要转义（如"."）

报错注入

着重记忆floor(),updatexml(),extractvalue()三个函数会导致报错注入

Windows下搜索不知名文件

将不可知字符用"<<"或">>"代替，"<",">"也可以，但只能代替一个，而前两者可以代替多个。

参数安全过滤

看懂第九章的三段代码

业务安全的解决方案

验证码

验证码绕过

防御：设置错误次数

不将验证码放在HTML或cookie

验证码请求一次后在后台强制刷新

短信轰炸

防御：限制同一个手机号，同一个IP一个时间段内请求接收短信的次数

登录

暴力破解（撞库）

防御：登录验证码与多因素认证

API登录

防御：采用随机登录秘钥

禁止搜索引擎收录API接口

登录密钥绑定主机

注册

防御：验证码

机器码拦截短时间多次注册

自学习识别垃圾账号

防止SQL与XSS

密码找回

防御：接收验证码的手机与邮箱不由用户控制

加强验证码复杂度

限制验证码错误次数

设置验证码失效时间

不将验证码保存在页面

取验证码时防止短信轰炸

验证码与用户名，邮箱绑定

查看，修改资料

防御：用户资源ID（如订单ID）绑定到用户

用户信息存储到session

投票 积分 抽奖

防御：机器码认证

操作需要登录，用户信息从session读取

充值支付

防御：保证数据可信，商品单价总价不从客户端获取

购买数量大于0

账户支付时锁定该账户，同时只能由一个请求操作

私信反馈

注意防XSS

远程地址访问

防御：限制填写内网与短地址

文件管理

防御：禁止写入脚本可在服务端执行的文件

限制文件管理功能操作的目录

限制文件管理功能访问权限

禁止上传特殊字符文件名的文件

数据库管理

防御：限制可以操作的数据库表

限制备份到服务器上的文件名

命令，代码执行

防御：严格控制该功能访问权限

设置命令白名单

设置独立密码

限制脚本可访问的路径

限制命令执行时的系统用户权限

文件、数据库备份

防御：控制该功能访问权限

文件名随机生成

API

防御：访问权限控制，只有登录后/拥有密钥才能调用

防止敏感信息泄露

防止SQL注入等常规漏洞